**Look up table technique:**

In this technique, we construct a 2d-array related to a particular query(Ex: Minimum value index in a given range(i,j)).

The size of the matrix is n\*n where(n is the number of elements).

The value 'k' at the index arr[i][j] represents that element at index k is minimum in the (range i to j)

Asymptotic analysis of look-up table technique

The space complexity of this algorithm is O(n^2)

The time complexity for construction is O(n^2)

Time complexity to query is O(1)

**Square root decomposition**

In this method, we construct a 1d-array whose is size is equal to square root of n.

Here each entry in the array stores the range query value.

Asymptoic analysis of square root decomposition

space complexity O(n)

The time complexity for construction O(n)

The time complexity for query O(1)

**Segment trees:**

Segment trees are complete binary trees which store the range query, and elements of the array are stored in leaf nodes

Asymptotic analysis of segment trees

space complexity O(2n+1)

The time complexity for construction O(n log)

The time complexity for query O(logn)

***Which one to choose?***

*If the element values don't change and Number of elements remain same then go with priority as follows*

1)segment trees

2)square root decomposition

3)Look up table

*If element value changes and number of elements remain same then go with priority as follows*

1)square root decomposition

2)Look up table

3)segment trees

*If the number of elements change then go with priority as follows*

1)lookup table because you need to recompute only the (x)colum of value.where x is the number of elements added.

2)square root decomposition because if only a few elements are added then the number of array elements does not change but index range changes

3)segment trees.

***Conclusion:***

There always a space-time complexity in the algorithm.

If the application needs speed and if it can afford with space then we go look up table method given the changes are rarely expected

If you cannot afford space then go with segment trees.